Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 65 (2015) 125 – 132

Procedia Computer Science

International Conference on Communication, Management and Information Technology (ICCMIT 2015)

Kinect 3D Point Cloud Live Video Streaming

Zainab Namh Sultani[1], Rana Fareed Ghani

Computer Science Department, University of Technology, Baghdad, Iraq

Abstract

We present a live video streaming system using a low cost 3D sensor camera like the Microsoft Kinect. The huge amount of raw point data that Kinect creates has to be stored and transmitted by efficient compact means. Noise and redundancy, however, make the process more difficult to achieve. To overcome these difficulties we propose a live streaming system that streams a 3D video to an Android Mobile phone and to Linux Desktop systems. For Android mobile phone client, the 3D video is filtered before streaming. Filtering stage contains 3 types of filters, Voxel Grid, Statistical Outlier Removal and Histogram-based conditional filters. The video is captured by the Kinect and a 3D point cloud is created. Voxel Grid filter is used since the generated 3D video will have millions of points; a downsampling procedure is applied to minimize the number of points. To reduce outliers and color information, statistical outlier removal and histogram-based conditional filters are used respectively. Conditional filter is customized by the scene histogram for each channel of RGB color to preserve the dominant color information for each scene. For Linux desktop client, the video is filtered by the histogram-based conditional filter and is compressed using an Octree structure that reduces spatial redundancies across the streamed video.

Peer-review under responsibility of Universal Society for Applied Research

Keywords: 3D Point Cloud, Kinect, Compression, Filter, Streaming

1. Introduction

Thanks to the great advancement of hardware, software, and algorithms in the past decade, our daily life has become a major digital content producer. Therefore the way we interact and communicate with machines have changed significantly as technology evolves new methods of interacting with computers. Classical means of remote communication are telephones, and more recently video conferencing systems such as Skype. While video conferencing systems enhanced the communication experience by incorporate video (visual) information in addition

1

* Corresponding author. *E-mail address:* zainab.sultani@informatik.uni-halle.de

Peer-review under responsibility of Universal Society for Applied Research
doi:10.1016/j.procs.2015.09.090

to audio, though it suffers from number of limitations: the speaker should maintain in front of the camera and these systems don't provide any 3D information [1].

Capturing 3D models for real-world scenes has been an active research topic for decades, with wide applications in gaming, video conferencing etc. With the brilliant design and the commoditization of 3D depth sensors, it has become easy to digitize the world into 3D models with millions of points. 3D point cloud data usually occupy a large amount of storage space, and demand efficient compression algorithms to store and transmit effectively [2].

To reduce the amount of information on 3D point clouds, several compression algorithms use the spatial organization of the points to encode them in a structure like an Octree. An Octree is a tree structure which contains a set of data called nodes, where each node can have up to eight child nodes, and so on recursively. Octrees make a partition of the three dimensional space by recursively subdividing it into eight subspaces. Another similar structure to the Octree approach is the Voxel Grid (VG). The VG downsampling technique is based on a grid of 3D box voxels which is used to reduce the number of points [3] [4] [5].

In Schnabel and Klein [6], a lossless progressive compression method is presented where the point cloud is encoded in terms of occupied octree-cells. The point cloud is compressed using a novel method of position and color prediction of the children nodes points of each Octree cell based on a local surface approximations.

Kammerl et al. [7] propose a spatially and temporal lossy compression of a point cloud stream. They have used Octrees to compress point clouds by performing a spatial decomposition. In order to detect and remove temporal redundancy, the difference between consecutive point clouds are considered. They have presented a technique for comparing the Octree data structure of their representation. Once compared, only the changes of the data are coded and sent to the decoder. This approach shows a strong compression ratio of 40x for controlled scenarios where there are only a few changes between different frames.

Following this idea of temporal redundancy between consecutive frames; Fu et al. [8] use the traditional video encoding method but with alteration to compress the depth images of Kinect devices. They use reference frames and use a depth prediction technique to generate consecutive frames. The results of this lossy compression report a compression of 55-85%.

Zhang et al. [2] present a graph transform to compress 3D point clouds attributes such as color. Graphs on small neighborhoods of the point cloud are constructed, by connecting nearby points, and treat the attributes as signals over the graph. Then a graph transform is applied on the generated graph to decorrelate the signal, which is equivalent to Karhunen-Loeve Transform.

Morell et al. [3] try to reduce the amount of data of a 3D point set but trying to preserve as much information as possible. The proposed method includes detecting planes, representing them with its plane equation, and extracting the points belonging to that plane. To preserve color information of those points, a color segmentation method is applied. The border of the group is extracted, obtaining a set of points defining a (concave) hull. Using triangles to decompose the surface of the plane, a Delaunay triangulation of the hull points is applied. So the information are: triangles with their vertex, distribution of the points, and color information. An initial compression ratio can be used to control the suggested method: when the number of processed points is over the ratio, the method can be stopped. The initial coordinates and color information are stored for the rest of points in the initial point set.

In this paper, we have designed two streaming systems, where a Kinect 3D live video is streamed to two different clients. An Android mobile phone client receives a filtered 3D video in real time, where the video is filtered using three types of filters. While a Linux Desktop client receives a filtered and compressed real time 3D video, where one type of filter is used.

The reminder of the paper provides a brief introduction about Microsoft Kinect and Point Cloud Library in sub-sections 1.1 and 1.2 respectively. Section 2 and Section 3 describe the suggested system for both clients. Section 4 and 5 represent the experimental results and the conclusion for the suggested systems.

1.1. Kinect 3D Camera

One of the most creative developments is the Microsoft Kinect camera. Kinect provides a great diversity of options for those looking to enrich their ability to work with computers. Kinect supports an RGB camera, multi-

array microphones, and a depth sensor capable of full-body 3D motion capture along with the facial and voice recognition capabilities [9], see Fig. 1.

Microsoft Kinect true power lies within its depth sensing technology. An infrared IR cluster of light (also known as point cloud) is produced and spread out across the room carrying information in the form of varying light patterns [10]. Kinect provides 3D scene using RGB and Depth data, where they are captured by RGB and IR cameras.

RGB-D cameras like Kinect provide useful data, which consist of 3D point clouds with color information. A point cloud is a data structure used to represent a collection of multi-dimensional points and is commonly used to represent three-dimensional data. In a 3D point cloud, the points usually hold the X, Y, and Z geometric coordinates. If the color information is presented, then the point cloud becomes 4D [11].

Fig. 1. Kinect 3D Camera.

1.2 Point Cloud Library (PCL)

The Point Cloud Library (PCL) [11] is a standalone, large scale, open project for 2D/3D image and point cloud processing. The PCL framework contains several state-of-the art algorithms including downsampling, filtering, feature estimation, surface reconstruction. It is free for commercial and research use, though it is released under the terms of the BSD license. The proposed system was implemented using PCL.

2. 3D Live Video Streaming System – Android Mobile Phone Client

The point cloud produced from Kinect sensor is dense and updated at real-time frame rate (30 fps). At normal case, 240k to 270k points are generated per second, which depend on the scene complexity. Therefore the dynamic point clouds contain a total of about 7.5 million points per second, which is a huge amount of data [12]. Any digital video has two types of redundancies: spatial redundancy and temporal redundancy. In order to decrease these redundancies different approaches are applied. Spatial redundancy is defined as repeated information inside the video frame as within an image, such as repeated pixel values. While temporal redundancy exists between consecutive frames where repeated information exist between them such as unchanged object position, or fixed background.

The streaming system was designed based on mobile server and client implementations [13] as the infrastructure of our work, though they have only used a Voxel grid filter. While in our approach, we have used three types of filters. In order to reduce the number of points in a point cloud data, a downsampling filter is used. First, the 3D point cloud video frame is divided using a Voxel grid, where the points within each Voxel grid are replaced (approximated) by the centroid of the 3D points within each Voxel.

After downsampling the 3D Point Cloud video, second filter is used to smooth the data points from outliers. Points that are statistically inconsistent with the rest of point cloud data are removed using an Outlier removal [14]. Outliers are removed by Statistical Outliers removal filter using mean and standard deviation. The mean distance is computed to all neighbors for each point. All points whose mean distances are outside an interval defined by the global distances mean and standard deviation can be considered as outliers and trimmed from the dataset.

Third filter is used since the 3D point cloud video data contains redundant color information, a Histogram-based Conditional filter will be used. A color histogram is a representation of an image colors distribution. For digital

images, a color histogram represents the number of pixels (number of occurrences) that have same color value in each of a fixed list of color ranges [15].

This filter will depend on the video frame histogram in order to capture the dominating color values for each RGB channel. Using these values, the point cloud video will filter out all values except the values used in the condition statements. See Fig. 2 for the proposed filtering stages.

After filtering, the 3D point cloud data will be sent over a client-server network, where the server is a Linux desktop and the client is an Android smartphone mobile. An Android mobile phone will receive the streamed point clouds over a TCP socket and render them using the VES and Kiwi mobile visualization framework.

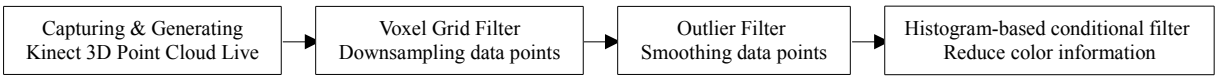

Fig. 2. Proposed Filtering Stages for Streaming to an Android Client

3. 3D Live Video Streaming System – Linux Desktop Client

Point clouds can be created at high rate and consequently will occupy a significant amount of memory resources. Once point clouds have to be stored or transmitted over rate-limited communication channels, methods for compressing this kind of data become highly important.

The Point Cloud Library provides point cloud compression functionality, where an Octree based compression class was designed to enable both spatial and temporal compression at the same time, see Fig. 3. The point cloud is constructed by nodes using Octree structure, where every non-empty node is occupied by set of points. Every non-empty node is subdivided into 8 octants, and the whole structure is encoded into byte stream. Temporal redundancy is measured by the exploiting the difference between consecutive point clouds, for more details see [7, 12].

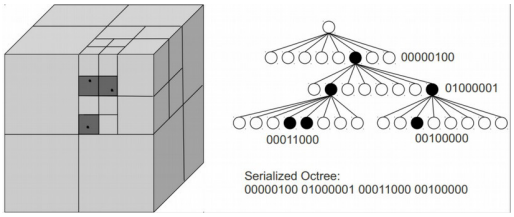

Fig. 3. Octree Structure.

In [16], the 3D point cloud was compressed before streaming to a client server network. In this paper, we have added a filtering stage before compression, see Fig.4. The 3D point cloud is filtered by histogram-based conditional filter before compressing and streaming. The conditional filter was designed using the same principle as mentioned above, where a frame color histogram is used to remove the less dominating colors (less occurrences).

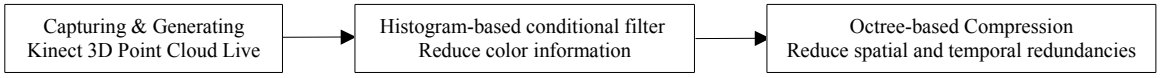

Fig. 4. Proposed Filter Stage for Streaming to a Linux Client

4. Experimental Results

4.1. Android Mobile Phone Client

The system was designed on a Linux operating system, with 3GHz CPU and 7.8 Gbyte RAM as the Server. Original 3D point cloud data is shown in Fig. 5. Number of points per frame of 3D point cloud before downsampling was 307,200 data points. Number of data points reduces after downsampling using voxel grid.

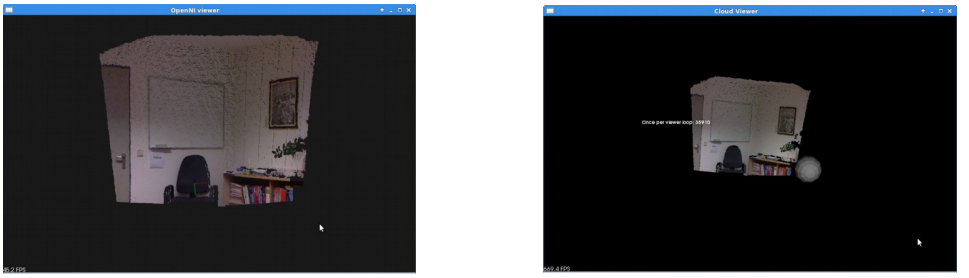

Fig. 5. (a) Left is the original 3D Point Cloud ; (b)Right is the Downsampled Point Cloud.

Statistical outlier removal and histogram-based conditional filters are applied after downsampling. Fig.6 shows the 3D point cloud after smoothing from outliers.

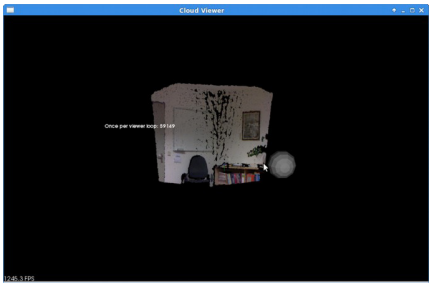

Fig. 6. 3D Point Cloud after outlier removal.

Using conditional filter reduces the number of points, if we do not use the *setKeepOrganized* function. Fig. 7 shows the color histogram for a 3D point cloud frame. Color histogram provides statistical information regarding dominating color values.

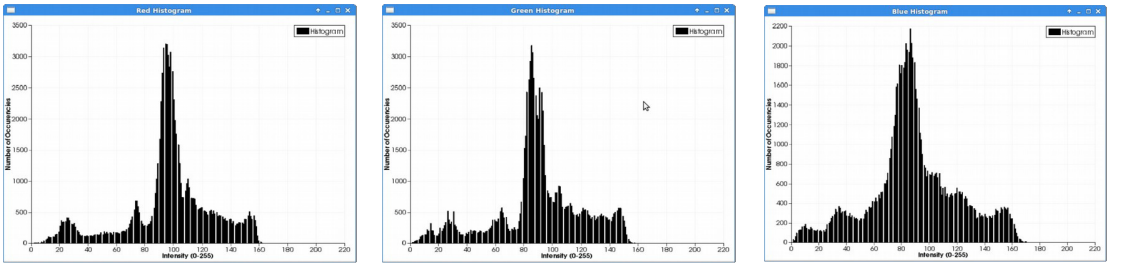

Fig. 7. (a). Left is the Red color Histogram ; (b). Middle is the Green Histogram ; (c). Right is the Blue Histogram

Fig. 8 represents the 3D point cloud frame after applying the histogram-based conditional filter.

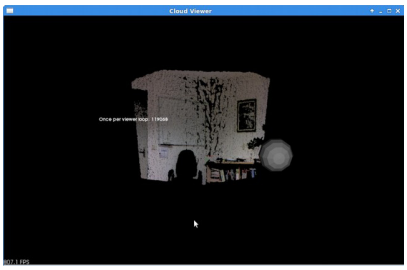

Fig. 8. 3D Point Cloud after histogram-based conditional filter

The filtered 3D point cloud video is ready to be live streamed to an Android mobile. Fig.9 shows the live streamed Kinect video from the server to Android client respectively.

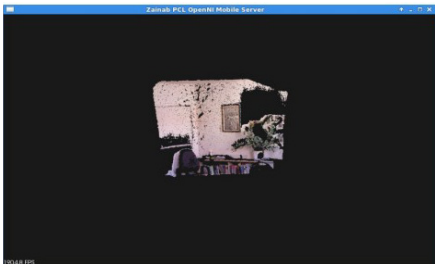

Fig. 9. (a). Left shows the streamed 3D point cloud from Server ; (b). Right shows the received 3D Point Cloud at Android mobile.

Table 1 displays the statistical information regarding the filtering impact on data points of the proposed system. Table 2 represents the statistical information of [13], where only Voxel Grid filter was used.

Table 1. Android Client Streaming Information - Proposed System.

Frames	Raw Data Points	Data Points After Downsampling	Data Points After Outlier removal	Data Points After Histogram-based conditional filter	Compression Ratio
A	307200	94596	84453	60986	5.037
B	307200	94500	90871	67034	4.582
C	307200	94675	84287	59960	5.123
D	307200	94262	83905	59706	5.145

Table 2. Android Client Streaming Information - [13].

Frames	Data Points After Downsampling	Compression Ratio
A	102246	3
B	102636	2.99
C	103007	2.982
D	101975	3.012

From Table 1 and 2, it can be seen that the outlier removal filter minimizes the number of data points in the 3D point cloud frame and therefore yields to better compression ratio values.

4.2. Linux Desktop Client

Using histogram-based conditional filter and Octree compression, the 3D point cloud is streamed to Linux desktop client. Fig. 10 represents the color histogram for each channel. Based on the most occurrence color range, the conditional filter was built. Fig. 11 (a) represents the received point cloud, if 50-240, 130-230 and 30-230 are used as the condition statements for Red, Green and Blue respectively. While in Fig. 11 (b), the condition statements are altered by using R: 50-240, G: 100-240 and B: 30-240, and it shows that the point cloud appearance is enhanced using these values.

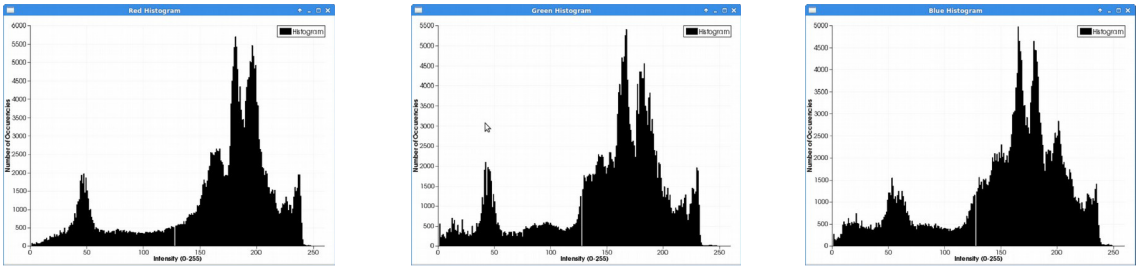

Fig. 10. (a). Left is the Red color Histogram ; (b). Middle is the Green Histogram ; (c). Right is the Blue Histogram.

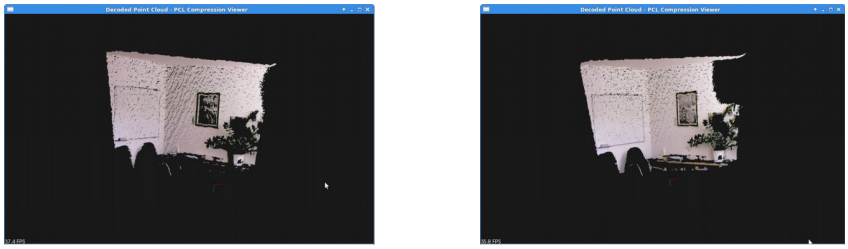

Fig. 11. (a). Left shows the streamed 3D point cloud at Linux Desktop; (b). Right shows the received 3D Point Cloud at Linux Desktop after color enhancement.

Table 3 displays the statistical information gathered during the proposed live streaming transmission between client server networks, where the size of compressed point cloud is 4500 kBytes. Table 4 gathered for system [16].

Table 3. Linux Desktop Client Streaming Information - Proposed System.

Frames	Size of compressed Point Cloud in kBytes	Total Bytes per Points in bytes	Total Compression Percentage	Compression Ratio
A	458.38	1.5279	10.186%	9.82
B	425.48	1.483	9.455%	10.58
C	424.62	1.415	9.4360	10.6

Table 4. Linux Desktop Client Streaming Information - [16].

Frames	Size of compressed Point Cloud in kBytes	Total Bytes per Points in bytes	Total Compression Percentage	Compression Ratio
A	473.67	1.5722	10.481%	9.51
B	472.92	1.5764	10.509%	9.52
C	473.95	1.5798	10.53%	9.49

5. Conclusion and Future Work

In this paper we have presented a live 3D video streaming system. A 3D point cloud video was captured using Microsoft Kinect camera. The generated 3D point cloud video was exposed to different filters and streamed to different systems. In Android client system, three types of filters were used. The data points were reduced using Voxel grid downsampling filter. A statistical outlier removal based on mean and standard deviation was applied to remove outliers. A conditional filter was designed based on the frame color histogram, to reduce the color information and to remove all the points that do not meet the conditional statements. Using outlier and histogram-based conditional filters enhanced the compression ratio. In Linux desktop client, the point cloud was filtered using histogram-based conditional filter and compressed using an Octree to reduce the temporal redundancy. Adding conditional filter reduced the size of compressed point clouds and therefore enhanced the compression ratio. Using compression increased the performance since the temporal redundancy was considered.

The designed system can be enhanced using extra information about data points, and not only depending on color information in designing conditional filters. From network perspective, the proposed system can be enhanced using different protocol that requires less overhead size such as UDP. Using UDP will stream the video faster, but it doesn't have recovery or re-transmission techniques like TCP. For fast streaming, a compromise should be made between accuracy and time.

References

1. Kuster C, Ranieri N, Agustina A, Zimmer H, Bazin J.C, Sun C, Popa T, Gross, M. *Towards next generation 3D teleconferencing systems. 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*; 2012.
2. Zhang C, Florencio D, Loop C. Point Cloud Attribute Compression with Graph Transform. *IEEE – Institute of Electrical and Electronics Engineers*; October 2014.
3. Morell V, Orts S, Cazorla M, Garcia-Rodriguez J. Geometric 3D point cloud compression. *Pattern Recognition Letters*; 2014, doi: http://dx.doi.org/10.1016/j.patrec.2014.05.016.
4. Connolly C.I. Cumulative generation of octree models from range data, in: *Proceedings of the First International Conference on Robotics and Automation, IEEE*; 1984, p. 25.
5. Kobbelt L, Botsch M. A survey of point-based techniques in computer graphics. *Computers & Graphics 28*; 2004.
6. Schnabel R, Klein R. Octree-based point-cloud compression, in: Botsch, M., Chen, B. (Eds.), *Symposium on Point-Based Graphics 2006,* Eurographics.
7. Kammerl J, Blodow N, Rusu R, Gedikli S, Beetz M, Steinbach E. Real-time compression of point cloud streams, in: *Robotics and Automation 235 (ICRA), 2012 IEEE International Conference*; 2012, p. 778–785.
8. Fu J, Miao D, Yu W, Wang S, Lu Y, Li S. Kinect-like depth compression with 2d+t prediction, in: *Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference*; 2012, p. 599–604.
9. Kramer J, Burrus N, Echtler F, Herrerac D, Parker M. Hacking the Kinect. Apress; 2012.
10. Jean S. Kinect Hacks. O'Reilly Median Inc.; 2013.
11. Rusu R.B, Cousins S. 3D is here: Point Cloud Library (PCL), in: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*; 2012, Shanghai, China.
12. Zhang L. TraumaBot - 3D body reconstruction & recognition; 2013. http://traumabot.blogspot.de/2013/06/octree-based-point-cloudstream.html.
13. Marion P. Point cloud streaming to mobile devices with real-time visualization; 2012. http://pointclouds.org/documentation/tutorials/mobile_streaming.php
14. Cimbala J. M. Instrumentation, Measurements, and Statistics lecture notes. Department of Mechanical and Nuclear Engineering at The Pennsylvania State University; 2014. http://www.mne.psu.edu/me345/lectures/Outliers.doc
15. Color Histogram – Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Color_histogram
16. Kammerl J. pcl openni organized compression. Willow Garage;2012. https://github.com/PointCloudLibrary/pcl/blob/master/apps/src/openni_organized_compression.cpp